PPG INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)
Recognized by UGC I Accredited by NAAC with A I ISO Certified Institution
NH 209, Sathy Main Road, Saravanampatti, Coimbatore – 641035, Tamil Nadu

CCS332 APP DEVELOPMENT

Department of

COMPUTER SCIENCE AND ENGINEERING

PPG INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)
Recognized by UGC I Accredited by NAAC with A I ISO Certified Institution
NH 209, Sathy Main Road, Saravanampatti, Coimbatore – 641035, Tamil Nadu

CCS345 APP DEVELOPMENT

NAME:..... ROLL NO: .....

SEMESTER: ..... BRANCH: .....

.....

Certified bonafide record of work done by**.....**

Place: COIMBATORE Date:

Staff In-Charge **Head of the Department**

University Register Number**: .....**

Submitted for the University Practical Examination held on**.....**

INTERNAL EXAMINER EXTERNAL EXAMINER

CONTENTS

Ex.No.1 Create a simple android application

Date:

Aim:

To create a simple hello android application.

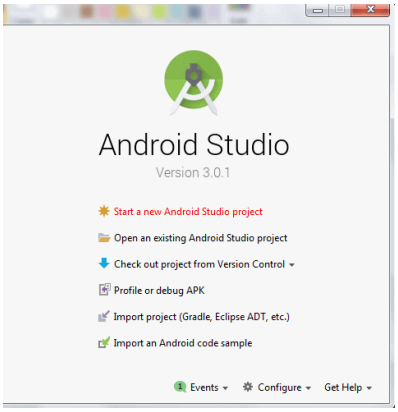
Procedure:

1. Create the new android project
2. Write the message (optional)
3. Run the android application

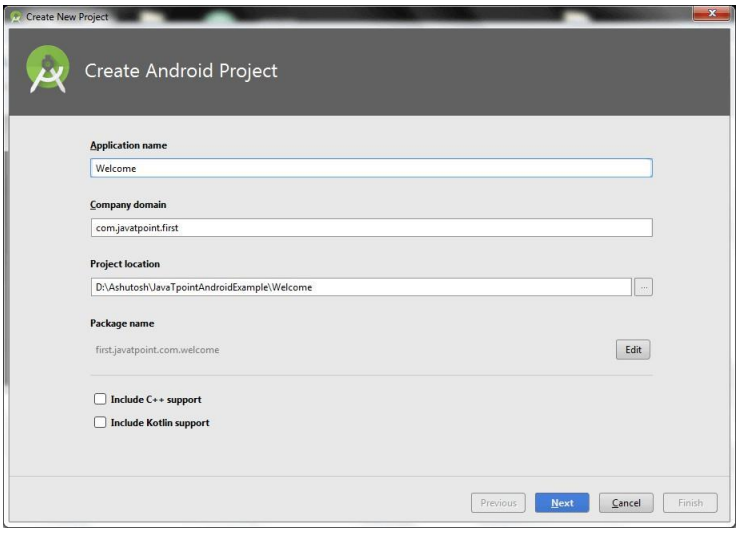
1) Create the New Android project

For creating the new android studio project:

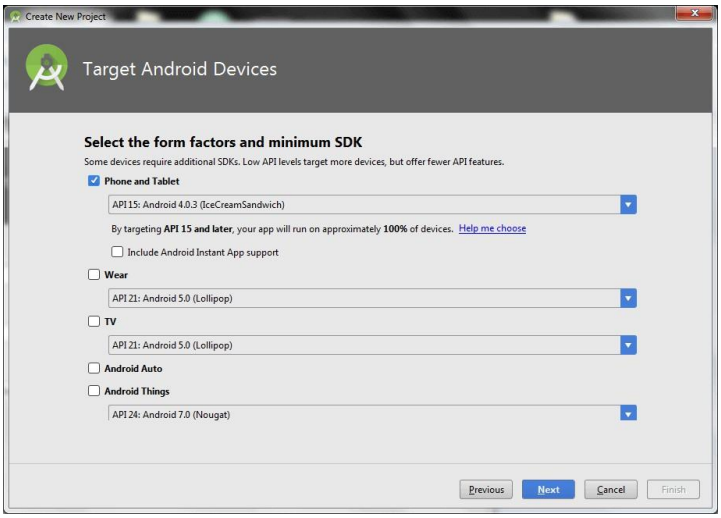
1) Select *Start a new Android Studio project*



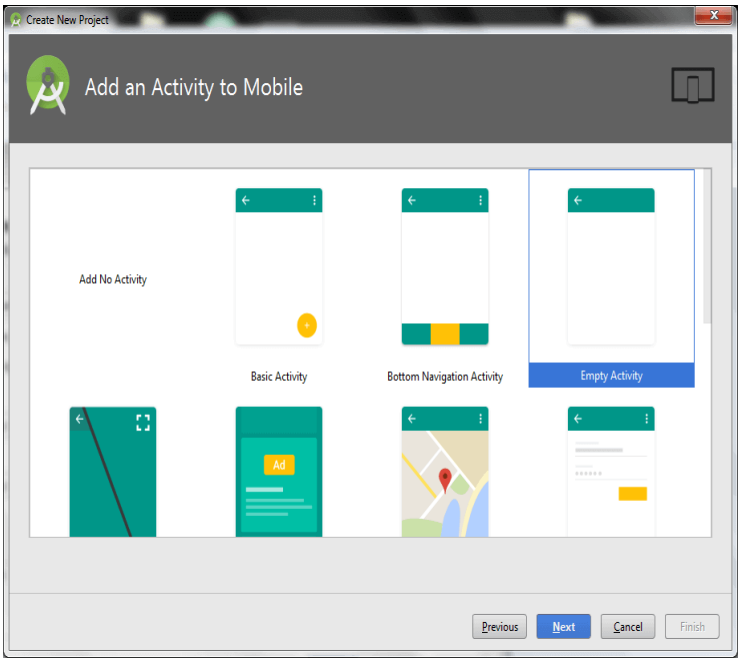
2) Provide the following information: Application name, Company domain, Project location and Package name of application and click next.



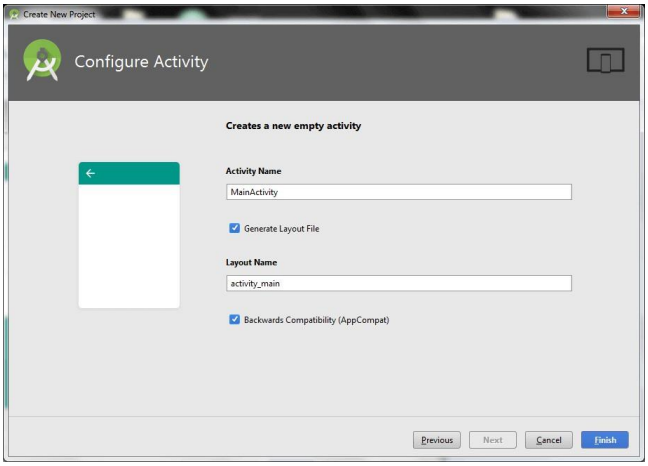
3) Select the API level of application and click next.



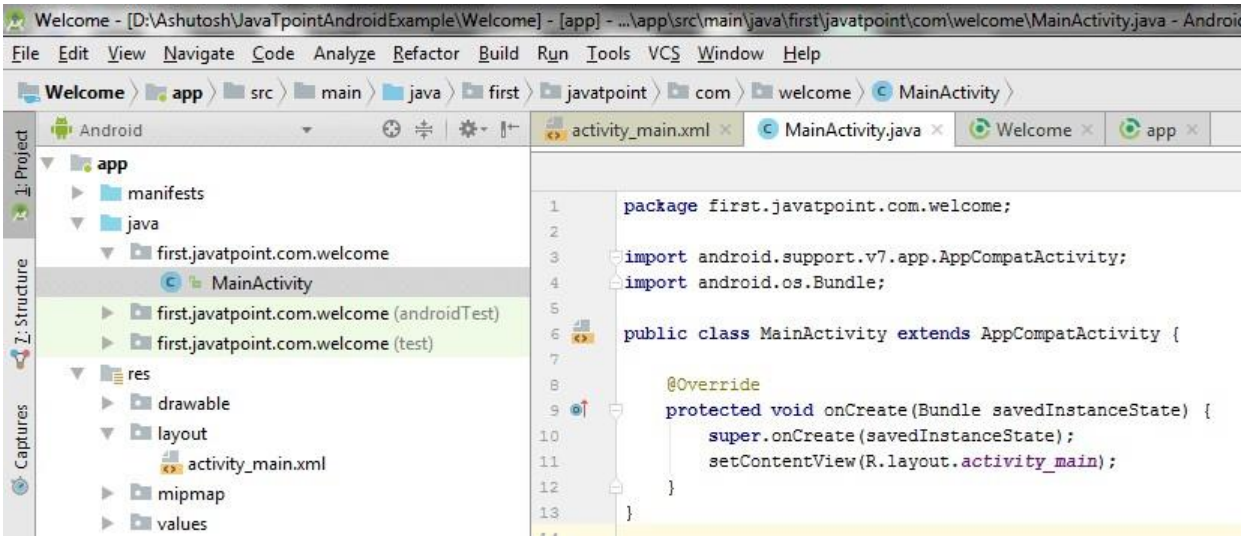
4) Select the Activity type (Empty Activity).



5) Provide the Activity Name and click finish.



- After finishing the Activity configuration, Android Studio auto generates the activity class and other required configuration files.
- Now an android project has been created. You can explore the android project and see the simple program, it looks like this:



2) Write the message

File: activity_main.xml

Android studio auto generates code for activity_main.xml file. You may edit this file according to your requirement.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://
schemas.android.co/apk/res    /android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="first.javatpoint.com.welcome.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

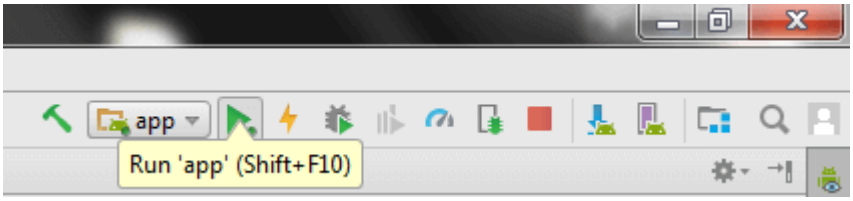
```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"  />

    </android.support.constraint.ConstraintLayout>
    }
```

File: MainActivity.java

```
package first.javatpoint.com.welcome;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

3) Run the android application

To run the android application, click the run icon on the toolbar or simply press Shift + F10.



The android emulator might take 2 or 3 minutes to boot. So please have patience. After booting the emulator, the android studio installs the application and launches the activity. You will see something like this:

Result:

Thus an android application has been created successfully .

EX.NO.2 GENERATING RANDOM NUMBER USING SEEKBAR

Date:

Aim:

To create a mobile application for generating random number using seek bar.

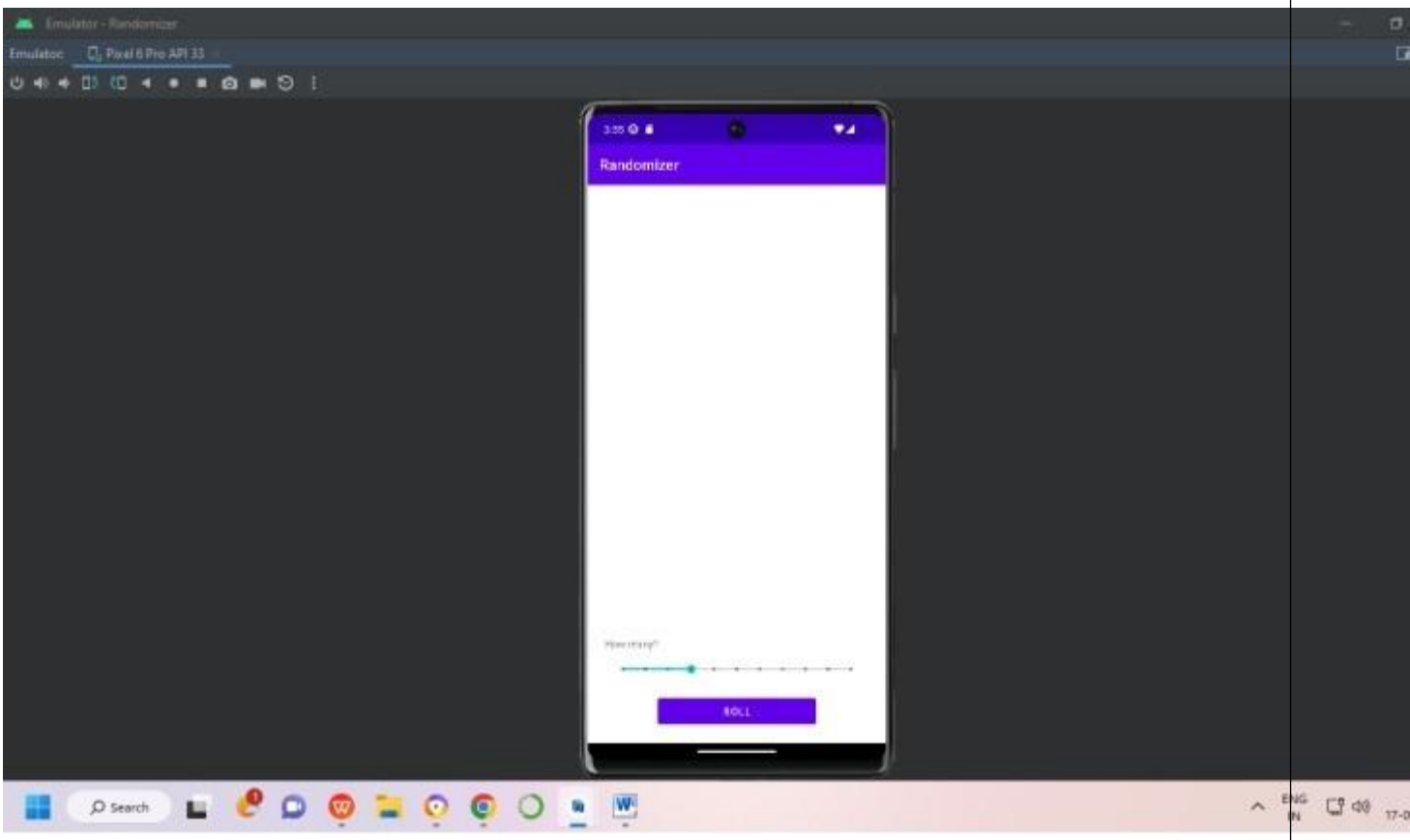
Procedure:

1. Create a New Android Project:
   - ➢ Select New in the toolbar and click New project
   - ➢ In the window that appears, select Empty Activity and then click Next
   - ➢ Provide the desired Activity name for your project, then select kotlin as language and click Finish.
2. Create a New AVD (Android Virtual Device):
   - ➢ Click Device Manager from the toolbar.
   - ➢ In the Device Manager panel, click create device.
   - ➢ Fill in the details for the AVD such as select phone from category, select pixel 6 pro then click Next.
   - ➢ Select tiramisu as system image.
     - ✓ If it is unavailable, Click tools select SDK manager, selct Android13.o(tiramisu) and then click apply and ok
     - ✓ Now the system image will be available.
   - ➢ Click Next and finish. Now click run to start the emulator.
3. Design the graphical layout with two text view, a command button , a horizontal divider and a seekbar.

   A. Change the id as following

      1. Button- id as rollButton and text as Roll.
      2. Seek bar.discrete- id as seekBar .
      3. TextView1- id as textView and text as How many?
      4. Divider discrete- id as divider
      5. TextView2 - id as resultsTextView and text size as 144sp and text as blank
4. Run the application.
5. On clicking the roll button random numbers are generated.
6. Seek bar is used to set the limitation for random number.
7. Close the Android project

Design



Program

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/rollButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="96dp"
        android:layout_marginEnd="96dp"
```

```
    android:layout_marginBottom="20dp"
    android:text="Roll"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
<SeekBar
    android:id="@+id/seekBar"
    style="@style/Widget.AppCompat.SeekBar.Discrete"
    android:layout_width="0dp"
    android:layout_height="19dp"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"
    android:layout_marginBottom="24dp"
    android:max="10"
    android:progress="3"
    app:layout_constraintBottom_toTopOf="@+id/rollButton"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginBottom="16dp"
    android:text="How many?"
    app:layout_constraintBottom_toTopOf="@+id/seekBar"
    app:layout_constraintStart_toStartOf="parent" />
<View
    android:id="@+id/divider"
    android:layout_width="409dp"
    android:layout_height="1dp"
    android:layout_marginBottom="16dp"
    android:background="?android:attr/listDivider"
    app:layout_constraintBottom_toTopOf="@+id/textView"
```

```
        tools:layout_editor_absoluteX="1dp" />
    <TextView
        android:id="@+id/resultsTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:textSize="144sp"
        app:layout_constraintBottom_toTopOf="@+id/divider"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.kt

```
package com.example.randomizer
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.SeekBar
import android.widget.TextView
import java.util.Random
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val rollButton=findViewById<Button>(R.id.rollButton)
        val resultsTextView=findViewById<TextView>(R.id.resultsTextView)
        val seekBar=findViewById<SeekBar>(R.id.seekBar)

        rollButton.setOnClickListener {
```

```
            val rand=Random().nextInt(seekBar.progress)
            resultsTextView.text=rand.toString()
        }
    }
}
```

Build.gradle:

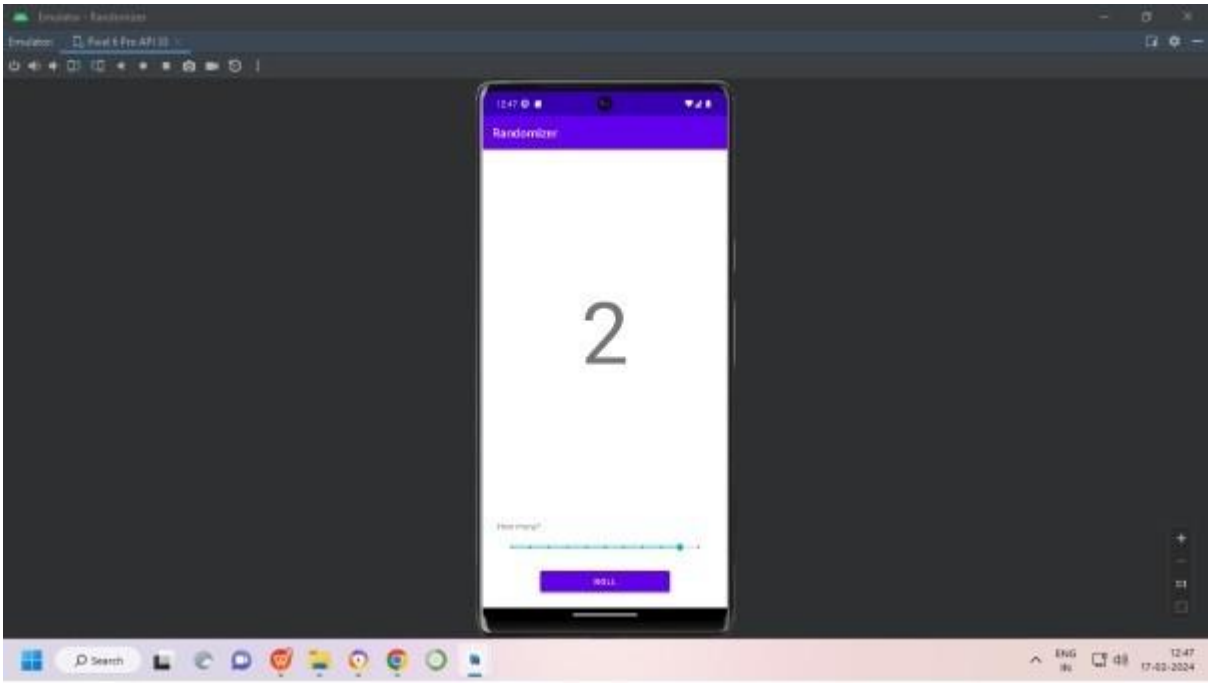
In build .gradle, change dependenies :material as 1.8.0,then click sync progress.

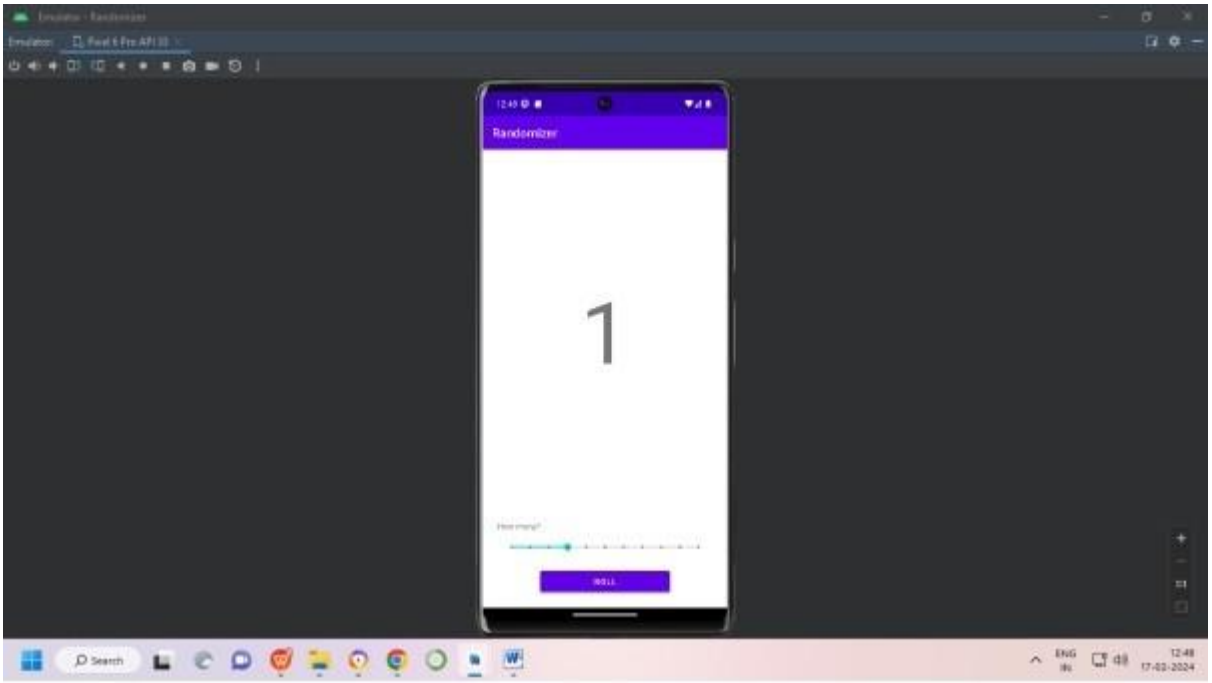
```
    implementation 'com.google.android.material:material:1.8.0'
```

code:

```
dependencies {
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation  'androidx.test.espresso:espresso-core:3.5.1'
}
```

output:





Result:

Thus an android application has been successfully created and outputs are verified.

Ex.No:3 CREATE PROJECTS IN REACT NATIVE

Date:

Aim:

To create first project in React Native.

Prerequisite:

1. Install Android studio.
2. Install Visual studio Code.
3. Install NodeJs.

Procedure:

Step1 : Open command prompt. Type the following

npx create-expo-app MyfirstProject

Step 2: Open Visual studio code. In that click Open folder and select MyfirstProject folder.

Step 3: Click App.js and edit the contents in <Text>.

Step 4:Open android Studio and run AVD device.

Step 5: In Visual studio code, Click Three dots present in the menu bar and select new terminal.

Step 5: In new terminal, type the following and press enter,

npx expo start

Step 6: Project will be successfully executed and options will be listed. In that press 'a' from the keyboard to run on android AVD.

Step7: Text will be displayed as output on Android. Make updates in text and save the content to view the changes reflected on android.

Step 8: Press ctrl+c to close the project.

Code:

App.js

```
import {View, Text} from "react-native";

export default function App(){
return(


    <View >

    <Text>Hurray! I got the Output!</Text>
    </View>

)
}
```

Run

PS C:\Users\CSE 29\Loginapp> npx expo start

Output

Result:

Thus the project has been successfully created in React Native and outputs are verified.

Ex.No:4 BMI CALCULATOR APP USING REACT NATIVE

Date:

Aim:

To Create a cross-platform BMI calculator app using React Native.

Procedure:

1. Install Node.js and verify the installation:

   Open command prompt and type the following commands to check the versions of Node.js and npm(Node Package Manager):

   - node -v
   - npm -v

2. Create a New React Native Project:

   Open command prompt and run the following command to create a new React Native project named "BMICalculatorApp":

   - npx create-expo-app BMICalculatorApp

3. Change the directory to the project folder:

   - cd BMICalculatorApp

4. Open the created project "BMICalculatorApp" in VS Code:

   Include dependencies for react-native-paper and @expo/vector-icons in package.json file.

5. Write the code for BMI Calculator in App.js file and save it.
6. Create a new AVD (Android Virtual Device) in Android Studio:
   - Click Device Manager from the toolbar.
   - In the Device Manager panel, click create device.
   - Fill in the details for the AVD such as select phone from category, select pixel 6 pro then click Next.
   - Select tiramisu as system image.
   - If it is unavailable, Click tools select SDK manager, select Android13.0(tiramisu) and then click apply and ok.
   - Now the system image will be available.
   - Click Next and finish. Now click run to start the emulator.
7. Type the following command in command prompt to run the application:

   - npm start

- Enter “a” to run the application in android device, it will start execution and display the app in AVD.

8. Enter Age, Height, Weight and Gender then click on “Calculate BMI”..
9. The calculated BMI is displayed along with Health Category.
10. Close the application.

Program:

App.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet } from 'react-native';
const App = () => {
 const [age, setAge] = useState('');
 const [height, setHeight] = useState('');
 const [weight, setWeight] = useState('');
 const [gender, setGender] = useState('');
 const [bmiResult, setBmiResult] = useState(null);
 const validateForm = () => {
  if (!age || !height || !weight || !gender) {
   alert('All fields are required!');
  } else {
   countBmi();
  }
 };
 const countBmi = () => {
  const bmi = (parseFloat(weight) / ((parseFloat(height) / 100) ** 2)).toFixed(2);
  let result = '';
  if (bmi < 18.5) {
   result = 'Underweight';
  } else if (bmi >= 18.5 && bmi <= 24.9) {
   result = 'Healthy';
  } else if (bmi >= 25 && bmi <= 29.9) {
   result = 'Overweight';
  } else if (bmi >= 30 && bmi <= 34.9) {
```

```
    result = 'Obese';
  } else if (bmi >= 35) {
    result = 'Extremely obese';
  }
  setBmiResult({ bmi, result });
  setAge('');
  setHeight('');

  setWeight('');
  setGender('');
};
return (
  <View style={styles.container}>
    {/* Input fields */}
    <TextInput
      placeholder="Age"
      value={age}
      onChangeText={(text) => setAge(text)}
      keyboardType="numeric"
      style={styles.input}
    />
    <TextInput
      placeholder="Height (cm)"
      value={height}
      onChangeText={(text) => setHeight(text)}
      keyboardType="numeric"
      style={styles.input}
    />
    <TextInput
      placeholder="Weight (kg)"
      value={weight}
      onChangeText={(text) => setWeight(text)}
      keyboardType="numeric"
      style={styles.input}
```

```
    />
    <TextInput
      placeholder="Gender"
      value={gender}
      onChangeText={(text) => setGender(text)}
      style={styles.input}
    />
    {/* Calculate button */}
    <TouchableOpacity onPress={validateForm} style={styles.calculateButton}>
      <Text style={styles.buttonText}>Calculate BMI</Text>
    </TouchableOpacity>
    {/* Display BMI result */}
    {bmiResult && (
      <View style={styles.resultContainer}>
        <Text>Your BMI: {bmiResult.bmi}</Text>
        <Text>Health Category: {bmiResult.result}</Text>
      </View>
    )}
  </View>
);
    };

    const styles = StyleSheet.create({
      container: {
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center',
        padding: 20,
      },
      input: {
        width: '80%',
        height: 40,
        borderWidth: 1,
        borderColor: '#ccc',
```

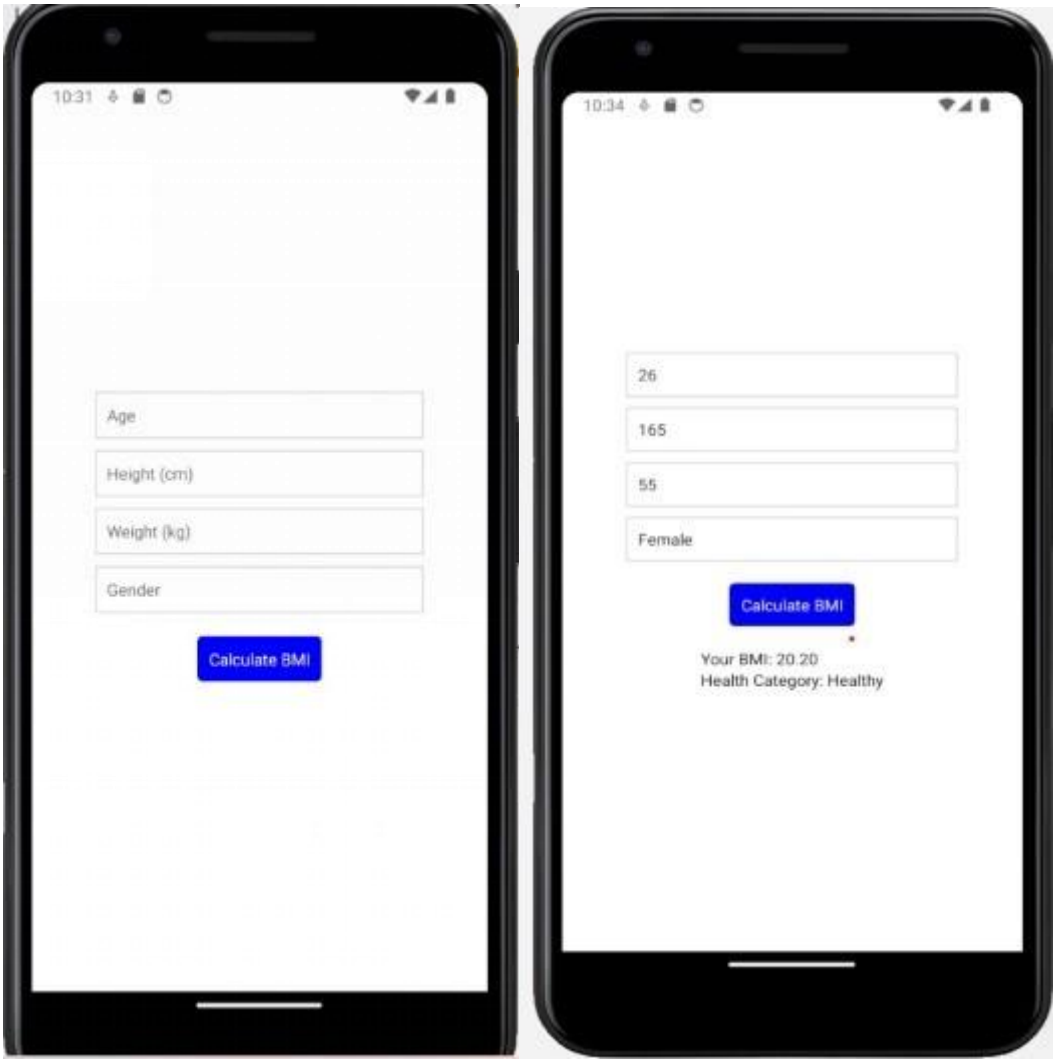
```
    marginBottom: 10,
    paddingHorizontal: 10,
  },
  calculateButton: {
    backgroundColor: 'blue',
    padding: 10,
    borderRadius: 5,
    marginTop: 10,
  },
  buttonText: {
    color: 'white',
    textAlign: 'center',
  },
  resultContainer: {
    marginTop: 20,
  },
});
export default App;
```

Package.json:

Include the following in the existing dependencies:

```
"react-native-paper": "4.9.2",
"@expo/vector-icons": "^13.0.0"
```

OUTPUT:



Result:

Thus the BMI Calculator Application using React Native has been created successfully and the output is verified.

EX.NO:5 EXPENSE MANAGER USING CROSS PLATFORM

Date:

Aim :

To build a cross platform application for a simple expense manager which allows entering expenses and income on each day and displays category wise weekly income and expense.

Introduction:

The Expense manager project assist users in keeping track of their expenses. With this app, users can easily add, and delete expenses to view a summary of their spending and it will show the available balance.

Procedure:

Step 1: Create a new React JS project using the following command

npx create-react-app exptrack

Step 2: Change to the project directory

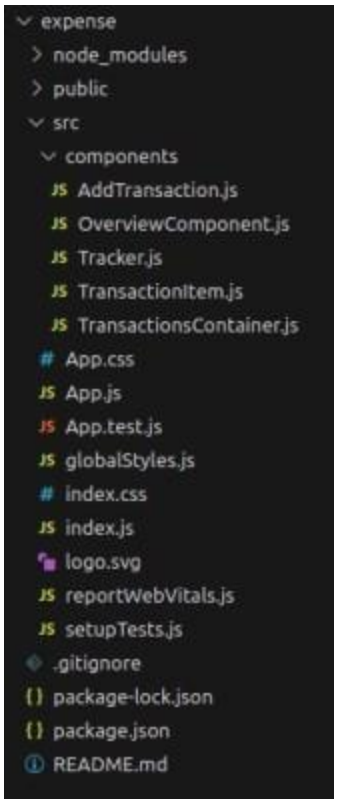
cd exptrack

Step 3: Install the requires modules

npm i styled-components

Step 4: Create a folder called components in src directory and create the following files inside it AddTransaction.js, OverviewComponent.js, Tracker.js, TransactionItem.js, TransactionsContainer.js and inside src directory create file globalStyles.js

Project Structure



Step 5:

The updated dependencies in **package.json** :

```
 "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "styled-components": "^6.0.8",
    "web-vitals": "^2.1.4"
  }
```

-

Code:

1. App.js: This component is responsible, for rendering the layout of the application.

```
// FileName: App.js
import styled from "styled-components";
import Tracker from "./components/Tracker";
import GlobalStyles from "./globalStyles";
const Main = styled.div`
display: flex;
justify-content: center;
align-items: center;
`;
const App = () => {
return (
    <Main>
    <GlobalStyles />
    <Tracker />
    </Main>
)
}
export default App;
```

2. **AddTransaction.js:** This component allows users to add transactions.

```
// FileName: AddTransaction.js

import { useState } from "react";
import styled from "styled-components";

const Container = styled.div`
text-align: center;
border: 1px solid #000;
padding: 20px;
border-radius: 5px;
```

```
margin-bottom: 25px;
`;
 const Input = styled.input`
width: 100%;
padding: 15px 20px;
outline: none;
border-radius: 5px;
margin: 5px 0;
border: 1px solid #000;
`;
 const RadioContainer = styled.div`
display: flex;
align-items: center;
justify-content: center;
`;
 const Label = styled.label`
margin-left: 10px;
cursor: pointer;
`;

const RadioBtn = styled(RadioContainer)`
margin: 10px 20px 10px 0;
`;
 const SubmitBtn = styled.button`
background-color: #44E610;
color: #fff;
border-radius: 5px;
padding: 10px 20px;
border: none;
outline: none;
cursor: pointer;
&:hover {
    background-color: #44E610;
}
```

```
`;

const AddTransaction = ({ setToggle, AddTransactions }) => {
const [amount, setAmount] = useState("");
const [details, setDetails] = useState("");
const [transType, setTransType] = useState("expense");

const AddTransactionData = () => {
    AddTransactions({
    amount: Number(amount),
    details,
    transType,
    id: Date.now(),
    });
    setToggle();
};

return (
    <Container>
    <Input
            type={"number"}
            placeholder="Enter Amount"
            value={amount}
            onChange={(e) => setAmount(e.target.value)}
    />

    <Input
            type={"text"}
            placeholder="Enter Details"
            value={details}
            onChange={(e) => setDetails(e.target.value)}
    />

    <RadioContainer>
```

```
            <RadioBtn>
            <input
                    type="radio"
                    id="expense"
                    name="type"
                    value={"expense"}
                    checked={transType === "expense"}
                    onChange={(e) => setTransType(e.target.value)}
            />
            <Label htmlFor="expense">Expense</Label>
            </RadioBtn>

            <RadioBtn>
            <input
                    type="radio"
                    id="income"
                    name="type"
                    value={"income"}
                    checked={transType === "income"}
                    onChange={(e) => setTransType(e.target.value)}
            />
            <Label htmlFor="income">Budget</Label>
            </RadioBtn>
    </RadioContainer>

    <SubmitBtn onClick={AddTransactionData}>Add Transaction</SubmitBtn>
    </Container>
);
};

export default AddTransaction;
```

3. OverviewComponent.js: This component displays the balance along with an “Add” button.

```
// FileName: OverviewComponent.js

import styled from "styled-components";

const Container = styled.div`
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 25px;
`;

const Balance = styled.h2`
font-weight: 500;
& span {
    font-weight: bold;
}
`;

const AddBtn = styled.button`
cursor: pointer;
background-color: rgb(68, 230, 16);
color: rgb(255, 255, 255);
padding: 7px 20px;
font-size: 16px;
border: none;
text-transform: uppercase;
border-radius: 5px;
`;

const OverviewComponent = ({ toggle, setToggle, income, expense }) => {
const bal = income - expense;
```

```
return (
    <Container>
    <Balance>
            Balance <span>₹{bal}</span>
    </Balance>
    <AddBtn onClick={() => setToggle(!toggle)}>
            {toggle ? "Cancel" : "Add"}
    </AddBtn>
    </Container>
);
};

export default OverviewComponent;
```

4. Tracker.js: The component that brings together parts of the application such as overview transaction list and addition of transactions.

```
// FileName: Tracker.js

import React, { useEffect, useState } from "react";
import styled from "styled-components";
import AddTransaction from "./AddTransaction";
import OverviewComponent from "./OverviewComponent";
import TransactionsContainer from "./TransactionsContainer";

const Container = styled.div`
display: flex;
flex-direction: column;
width: 600px;
max-width: 100%;
background-color: #fff;
padding: 30px 20px;
border: 1px solid #000;
```

```
border-radius: 5px;
margin: 10px;
`;

const Heading = styled.h1`
font-size: 30px;
font-weight: bold;
text-align: center;
margin-bottom: 20px;
`;

const TransactionDetails = styled.div`
display: flex;
justify-content: space-between;
align-items: center;
gap: 20px;
margin-bottom: 25px;
`;

const THeading = styled.div`
font-size: 30px;
font-weight: bold;
text-align: center;
margin-bottom: 20px;
color: #44E610;
`;

const ExpenseBox = styled.div`
flex: 1;
border: 1px solid #000;
border-radius: 5px;
padding: 10px 20px;
background-color: #fff;
& span {
```

```
    font-weight: bold;
    font-size: 25px;
    display: block;
    color: ${(props) => (props.isExpense ? "red" : "green")};
}
`;

const IncomeBox = styled(ExpenseBox)``;

const Tracker = () => {
const [toggle, setToggle] = useState(false);
const [transactions, setTransactions] = useState([]);
const [expense, setExpense] = useState(0);
const [income, setIncome] = useState(0);

const AddTransactions = (payload) => {
    const transactionArray = [...transactions];
    transactionArray.push(payload);
    setTransactions(transactionArray);
};

const removeTransaction = (id) => {
    const updatedTransactions = transactions
                                    .filter((transaction) => transaction.id !== id);
    setTransactions(updatedTransactions);
};

const calculateTransactions = () => {
    let exp = 0;
    let inc = 0;

    transactions.map((item) => {
    item.transType === "expense"
            ? (exp = exp + item.amount)
```

```
            : (inc = inc + item.amount);
    });

    setExpense(exp);
    setIncome(inc);
};

useEffect(() => {
    calculateTransactions();
}, [transactions]);

return (
    <Container>
            <THeading>GeeksforGeeks</THeading>
    <Heading>Expense Tracker</Heading>
    <OverviewComponent
            toggle={toggle}
            setToggle={setToggle}
            expense={expense}
            income={income}
    />

    {toggle && (
            <AddTransaction
            setToggle={setToggle}
            AddTransactions={AddTransactions}
            />
    )}

    <TransactionDetails>
            <ExpenseBox isExpense>
            Expense <span>₹{expense}</span>
            </ExpenseBox>
```

```
            <IncomeBox>
            Budget <span>₹{income}</span>
            </IncomeBox>
    </TransactionDetails>

    <TransactionsContainer
            transactions={transactions}
            removeTransaction={removeTransaction}
    />
    </Container>
);
};

export default Tracker;
```

5. TransactionItem.js: This component is responsible, for displaying transaction details including description, amount and a button to remove it from the list.

```
// FileName: TransactionItem.js

import React from "react";
import styled from "styled-components";

const Item = styled.div`
display: flex;
justify-content: space-between;
align-items: center;
border: 1px solid #e6e8e9;
background-color: #fff;
border-radius: 5px;
padding: 10px 20px;
border-right: 5px solid ${(props) => (props.isExpense ? "red" : "green")};
margin-bottom: 10px;
cursor: pointer;
```

```
`;

const RemoveButton = styled.button`
background-color: #44E610;
color: white;
border: none;
padding: 5px 10px;
border-radius: 3px;
cursor: pointer;
`;

const TransactionItem = ({ transaction, removeTransaction }) => {
return (
    <Item isExpense={transaction?.transType === "expense"}>
    <span>{transaction.details}</span>
    <span>₹{transaction.amount}</span>
    <RemoveButton onClick={() => removeTransaction(transaction.id)}>
            Remove
    </RemoveButton>
    </Item>
);
};

export default TransactionItem;
```

6. TransactionsContainer.js: This component. Filters the list of transactions. It offers a search input field and displays only filtered transaction items.

```
// FileName: TransactionsContainer.js
import React, { useEffect, useState } from "react";
import styled from "styled-components";
import TransactionItem from "./TransactionItem";

const Container = styled.div``;

const Heading = styled.h2`
```

```
font-size: 25px;
font-weight: 600;
`;

const SearchInput = styled.input`
width: 100%;
padding: 15px 20px;
outline: none;
border-radius: 5px;
margin: 5px 0;
border: 1px solid #e6e8e9;
background-color: #e6e8e9;
margin-bottom: 25px;
`;
 const TransactionItems = styled.div``;
 const TransactionsContainer = ({ transactions, removeTransaction }) => {
const [searchInput, setSearchInput] = useState("");
const [filteredTransactions, setFilteredTransactions] = useState(transactions);

const filteredData = (searchInput) => {
    if (!searchInput || !searchInput.trim().length) {
    setFilteredTransactions(transactions);
    return;
    }

    let filtered = [...filteredTransactions];
    filtered = filtered.filter(
    (item) =>
            item.details.toLowerCase().includes(searchInput.toLowerCase().trim())
    );
    setFilteredTransactions(filtered);
};

useEffect(() => {
```

```
      filteredData(searchInput);
 }, [transactions, searchInput]);

return (
    <Container>
    <Heading>Transactions</Heading>

    <SearchInput
            type="text"
            placeholder="Search here"
            value={searchInput}
            onChange={(e) => setSearchInput(e.target.value)}
    />

    <TransactionItems>
            {filteredTransactions?.length ? (
            filteredTransactions.map((transaction) => (
                    <TransactionItem
                    transaction={transaction}
                    key={transaction.id}
                    removeTransaction={removeTransaction}
                    />
            ))
            ) : (
            <p>No Transactions</p>
            )}
    </TransactionItems>
    </Container>
);
};

export default TransactionsContainer;
```

7. globalStyles.js

```
// FileName: globalStyles.js
import { createGlobalStyle } from "styled-components";
const GlobalStyles = createGlobalStyle`
    *{
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Poppins', sans-serif;
    }
`
export default GlobalStyles;
```

Steps to run the project:

Step 1: Type the following command in terminal.

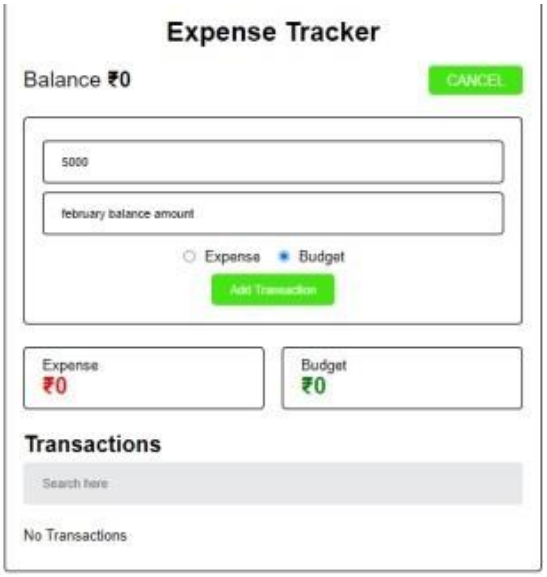
```
npm start
```

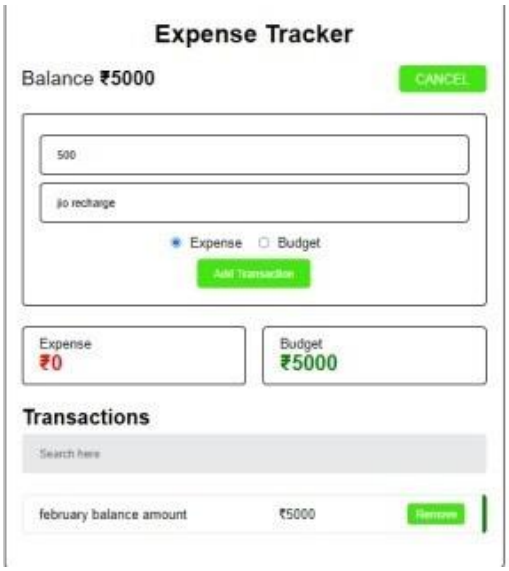
Step 2: Open web-browser and type the following URL

```
http://localhost:3000/
```

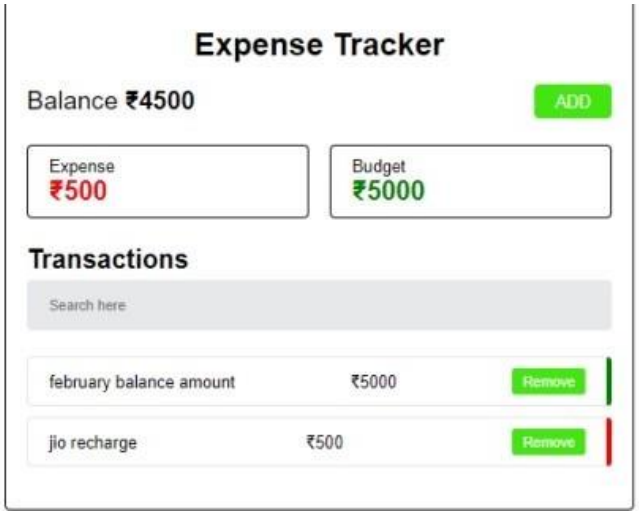
Output

1. BUDGET OUTPUT





2. EXPENSE OTPUT



Result:

Thus a cross platform application for a simple expense manager has been successfully created and the outputs are verified.

EX.NO.6 CONVERT UNITS FROM IMPERIAL SYSTEM TO METRIC SYSTEM

DATE:

Aim:

To develop a cross platform application to convert units from imperial system to metric system (km to miles, kg to pounds etc.,)

Procedure:

1. Install Node.js and verify the installation:

   Open command prompt and type the following commands to check the versions of Node.js and npm(Node Package Manager):

   - node -v
   - npm -v

2. Create a react application by using this command

   i) Open command prompt and run the following command to create a new React Native project named “LengthConverter”:

   npx create-react-app LengthConverter

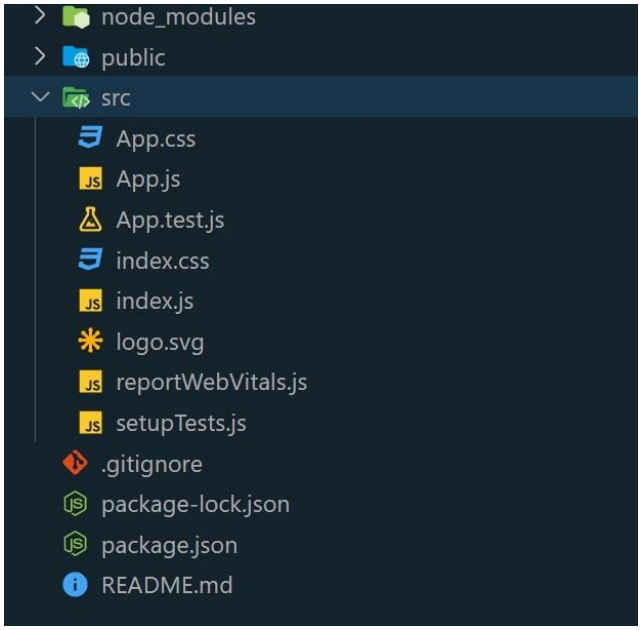
3. Change the directory to the project folder:

   - cd LengthConverter

4. Open the created project “BMICalculatorApp” in VS Code:

   Include dependencies for react-native-paper and @expo/vector-icons in package.json file.

5. Open Project Structure:



6. The updated dependencies in package.json will look like:

```
"dependencies": {
      "@testing-library/jest-dom": "^5.17.0",
       "@testing-library/react": "^13.4.0",
       "@testing-library/user-event": "^13.5.0",
       "react": "^18.2.0",
       "react-dom": "^18.2.0",
       "react-scripts": "5.0.1",
       "web-vitals": "^2.1.4"
}
```

7. Write the below code in App.js file and App.css in the src directory
   a) **App.js:** This component of our app contains all the logic
   b) **App.css:** This file contains the styling of our project
8. Create a new AVD (Android Virtual Device) in Android Studio:
   - Click Device Manager from the toolbar.
   - In the Device Manager panel, click create device.
   - Fill in the details for the AVD such as select phone from category, select pixel 6 pro then click Next.
   - Select tiramisu as system image.
   - If it is unavailable, Click tools select SDK manager, select Android13.0(tiramisu) and then click apply and ok.
   - Now the system image will be available.
   - Click Next and finish. Now click run to start the emulator.
9. Run the following command in the terminal and visit the given URL to see the output: **http://localhost:3000/**
   - npm start
10. Close the application.

Code:

App.js

```
import React, { useState } from "react";
import "./App.css";
function App() {
	const [inputValue, setInputValue] = useState(0);
	const [fromUnit, setFromUnit] = useState("cm");
	const [toUnit, setToUnit] = useState("inch");
	const [result, setResult] = useState(0);

	const conversionFactors = {
		cm: {
			inch: 1 / 2.54,
			feet: 1 / 30.48,
			meter: 1 / 100,
			yard: 1 / 91.44,
			mile: 1 / 160934.4,
			kilometer: 1 / 100000,
			micrometer: 1e4,
			nanometer: 1e7,
			millimeter: 10,
			nauticalMile: 1 / 1852,
			exameter: 1e-17,
			petameter: 1e-14,
			terameter: 1e-11,
		},
		inch: {
			cm: 2.54,
			feet: 1 / 12,
			meter: 0.0254,
			yard: 1 / 36,
			mile: 1 / 63360,
```

```
        kilometer: 0.0000254,
        micrometer: 25400,
        nanometer: 25400000,
        millimeter: 25.4,
        nauticalMile: 1 / 72913.4,
        exameter: 1e-16,
        petameter: 1e-13,
        terameter: 1e-10,
    },
    feet: {
        cm: 30.48,
        inch: 12,
        meter: 0.3048,
        yard: 1 / 3,
        mile: 1 / 5280,
        kilometer: 0.0003048,
        micrometer: 304800,
        nanometer: 304800000,
        millimeter: 304.8,
        nauticalMile: 1 / 6076.12,
        exameter: 1e-15,
        petameter: 1e-12,
        terameter: 1e-9,
    },
    meter: {
        cm: 100,
        inch: 39.3701,
        feet: 3.28084,
        yard: 1.09361,
        mile: 1 / 1609.34,
```

```
        kilometer: 0.001,
        micrometer: 1e6,
        nanometer: 1e9,
        millimeter: 1000,
        nauticalMile: 1 / 1852,
        exameter: 1e-18,
        petameter: 1e-15,
        terameter: 1e-12,
    },
    yard: {
        cm: 91.44,
        inch: 36,
        feet: 3,
        meter: 0.9144,
        mile: 1 / 1760,
        kilometer: 0.0009144,
        micrometer: 914400,
        nanometer: 914400000,
        millimeter: 914.4,
        nauticalMile: 1 / 2025.37,
        exameter: 1e-17,
        petameter: 1e-14,
        terameter: 1e-11,
    },
    mile: {
        cm: 160934.4,
        inch: 63360,
        feet: 5280,
        meter: 1609.34,
        yard: 1760,
```

```
        kilometer: 1.60934,
        micrometer: 1.60934e9,
        nanometer: 1.60934e12,
        millimeter: 1.60934e6,
        nauticalMile: 1 / 1.15078,
        exameter: 1e-15,
        petameter: 1e-12,
        terameter: 1e-9,
    },
    kilometer: {
        cm: 1e5,
        inch: 39370.1,
        feet: 3280.84,
        meter: 1000,
        yard: 1093.61,
        mile: 0.621371,
        micrometer: 1e9,
        nanometer: 1e12,
        millimeter: 1e6,
        nauticalMile: 0.539957,
        exameter: 1e-16,
        petameter: 1e-13,
        terameter: 1e-10,
    },
    micrometer: {
        cm: 1e-4,
        inch: 3.937e-5,
        feet: 3.2808e-6,
        meter: 1e-6,
        yard: 1.0936e-6,
```

```
        kilometer: 1e-9,
        nanometer: 1000,
        millimeter: 0.001,
        nauticalMile: 5.3996e-10,
        exameter: 1e-23,
        petameter: 1e-20,
        terameter: 1e-17,
    },
    nanometer: {
        cm: 1e-7,
        inch: 3.937e-8,
        feet: 3.2808e-9,
        meter: 1e-9,
        yard: 1.0936e-9,
        mile: 6.2137e-13,
        kilometer: 1e-12,
        micrometer: 1e-6,
        millimeter: 1e-6,
        nauticalMile: 5.3996e-13,
        exameter: 1e-26,
        petameter: 1e-23,
        terameter: 1e-20,
    },
    millimeter: {
        cm: 0.1,
        inch: 0.03937,
        feet: 0.003281,
        meter: 0.001,
        yard: 0.0010936,
```

```
        mile: 6.2137e-7,
        kilometer: 1e-6,
        micrometer: 1000,
        nanometer: 1e6,
        nauticalMile: 5.3996e-7,
        exameter: 1e-20,
        petameter: 1e-17,
        terameter: 1e-14,
    },
    nauticalMile: {
        cm: 185200,
        inch: 72913.4,
        feet: 6076.12,
        meter: 1852,
        yard: 2025.37,
        mile: 1.15078,
        kilometer: 1.852,
        micrometer: 1.852e8,
        nanometer: 1.852e11,
        millimeter: 1.852e5,
        exameter: 5.3996e-17,
        petameter: 5.3996e-14,
        terameter: 5.3996e-11,
    },
    exameter: {
        cm: 1e17,
        inch: 3.937e16,
        feet: 3.2808e16,
        meter: 1e16,
        yard: 1.0936e16,
```

```
        mile: 6.2137e12,
        kilometer: 1e15,
        micrometer: 1e23,
        nanometer: 1e26,
        millimeter: 1e20,
        nauticalMile: 5.3996e17,
        petameter: 1e3,
        terameter: 1e6,
    },
    petameter: {
        cm: 1e14,
        inch: 3.937e13,
        feet: 3.2808e13,
        meter: 1e13,
        yard: 1.0936e13,
        mile: 6.2137e9,
        kilometer: 1e12,
        micrometer: 1e20,
        nanometer: 1e23,
        millimeter: 1e17,
        nauticalMile: 5.3996e14,
        exameter: 1e-3,
        terameter: 1e3,
    },
    terameter: {
        cm: 1e11,
        inch: 3.937e10,
        feet: 3.2808e10,
        meter: 1e10,
        yard: 1.0936e10,
```

```
            mile: 6.2137e6,
            kilometer: 1e9,
            micrometer: 1e17,
            nanometer: 1e20,
            millimeter: 1e14,
            nauticalMile: 5.3996e11,
            exameter: 1e-6,
            petameter: 1e-3,
        },
};

const convert = () => {
        const convertedResult =
            inputValue * conversionFactors[fromUnit][toUnit];
        setResult(convertedResult.toFixed(4));
};

return (
        <div className="container">
            <h1>Length Converter</h1>
            <input
                type="number"
                value={inputValue}
                onChange={(e) => setInputValue(parseFloat(e.target.value))}
                placeholder="Enter value"
                className="input-field"
            />
            <select
                className="unit-selectors"
                value={fromUnit}
```

```
                        onChange={(e) => setFromUnit(e.target.value)}
                    >
                        {Object.keys(conversionFactors).map((unit) => (
                            <option key={unit} value={unit}>
                                {unit}
                            </option>
                        ))}
                    </select>
                    <select
                        className="unit-selectors"
                        value={toUnit}
                        onChange={(e) => setToUnit(e.target.value)}
                    >
                        {Object.keys(conversionFactors[fromUnit]).map((unit) => (
                            <option key={unit} value={unit}>
                                {unit}
                            </option>
                        ))}
                    </select>
                    <p id="result">{result}</p>
                    <button onClick={convert} className="convert-button">
                        Convert
                    </button>
                </div>
        );
}

export default App;
```

App.css

```
body {
background: #eee;
}
.container {
max-width: 600px;
margin: 30px auto;
background-color: white;
padding: 20px;
border-radius: 10px;
text-align: center;
box-shadow: 0 2px 6px rgba(0, 0, 0, 0.2);
}

.container h1 {
font-size: 24px;
margin-bottom: 20px;
color: #0074cc;
}

.input-field {
width: 90%;
padding: 10px;
margin: 10px 0;
border: 1px solid #ccc;
border-radius: 5px;
font-size: 16px;
}

.unit-selectors {
flex: 1;
```

```
padding: 10px;
border: 2px solid #ccc;
border-radius: 10px;
font-size: 16px;
margin: 10px;
}

#result {
font-size: 20px;
margin-top: 20px;
color: crimson;
}

.convert-button {
background-color: #0074cc;
color: white;
padding: 12px 24px;
border: none;
border-radius: 5px;
font-size: 18px;
cursor: pointer;
transition: background-color 0.3s;
}

.convert-button:hover {
background-color: #0056a4;
}
```

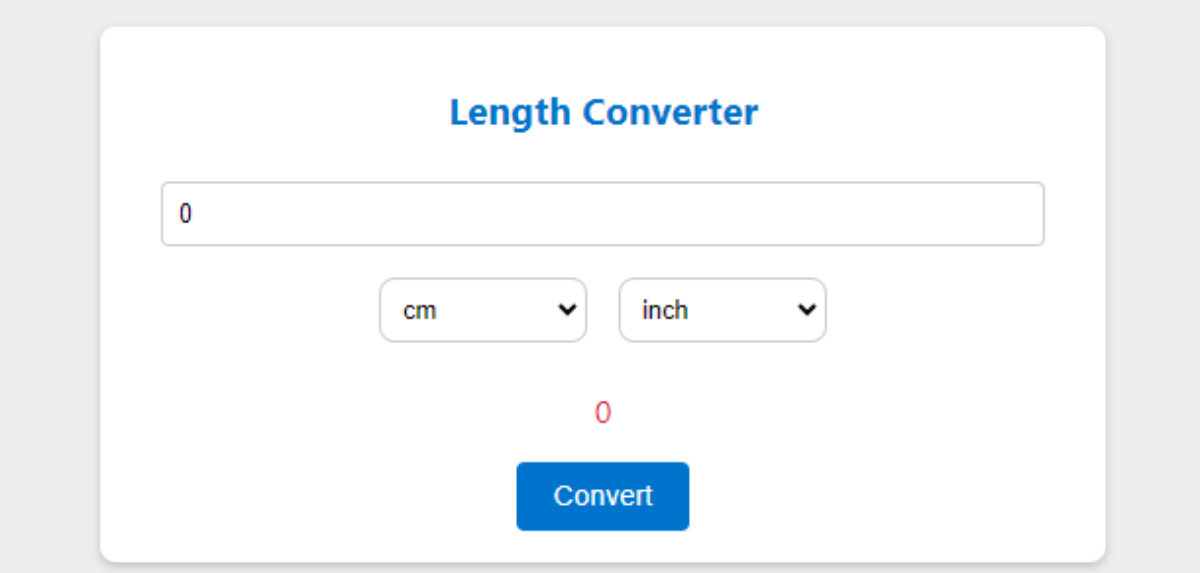
Execution:

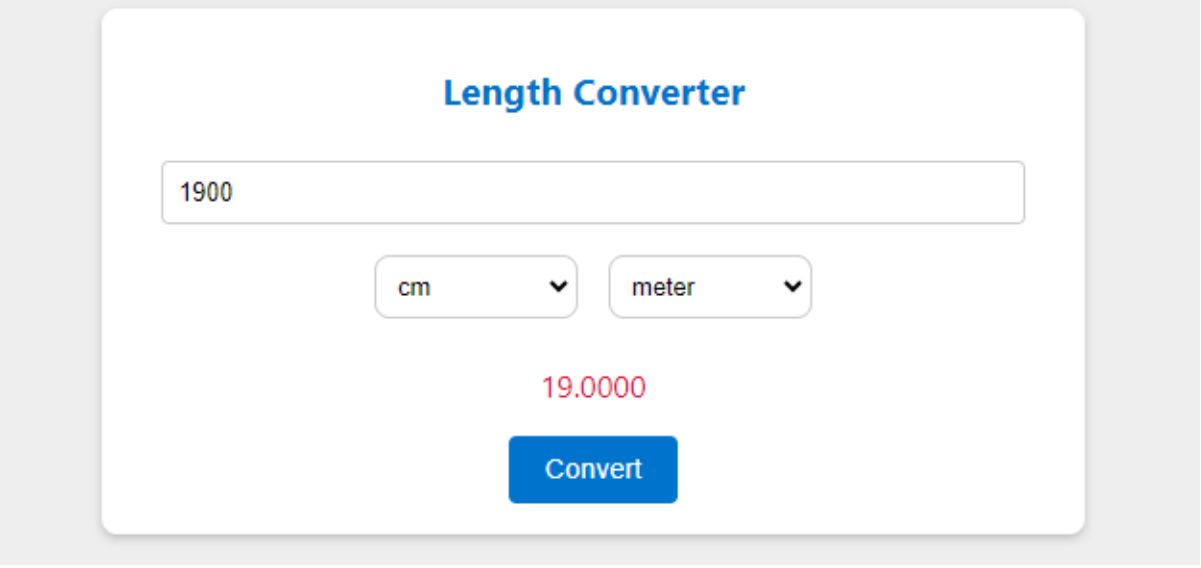
To run the Application:

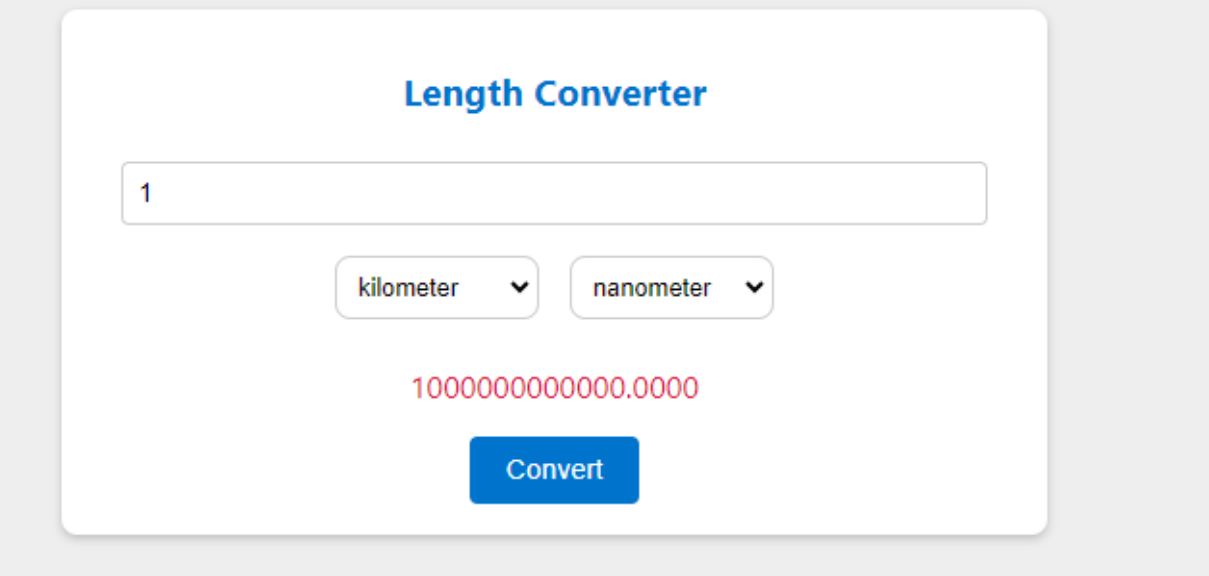
Run the following command in the terminal and visit the given URL to see the output: **http://localhost:3000/**

npm start

OUTPUT







Result:

Thus a cross platform application to convert units from imperial system to metric system has been successfully developed and outputs are verified.

EX.NO.7 A CROSS PLATFORM APPLICATION FOR DAY TO DAY TASK MANAGEMENT.

Date:

Aim:

To design and develop a cross platform application for day to day task (to-do) management.

Introduction:

The application enables users to effortlessly create, edit, complete/incomplete, and delete tasks, providing an uncomplicated mobile app capabilities.

Steps to install & configure React Native"

Step 1: Create a react native application by using this command:

npx create-expo-app taskmanagerapp

Step 2: After creating your project folder, i.e. taskmanagerapp, use the following command to navigate to it:

cd taskmanagerapp

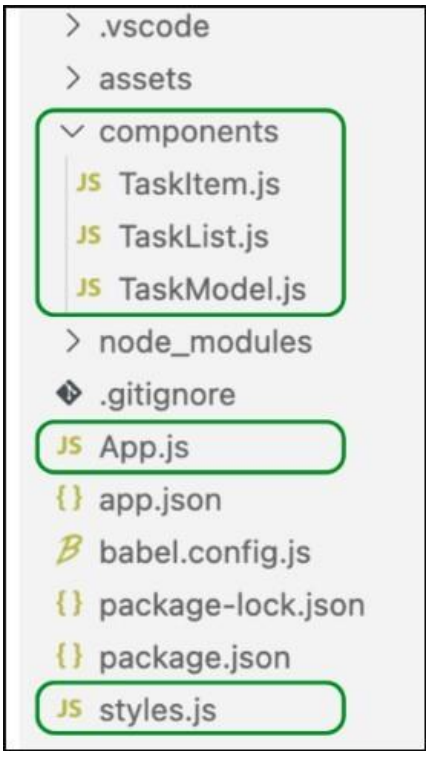
Step 3: Use an external library for a Date Picker

npm i react-native-modern-datepicker

Step 4: create new file as styles.js. Create a folder called components and include the following files

TaskItem.js,TaskList.js,TaskModel.js

Project structure



Code

i. TaskModal.js

```
// components/TaskModal.js
import React from "react";
import {
    View,
    Text,
    TextInput,
    Button,
```

```
    Modal,
} from "react-native";
import styles from "../styles";
import DatePicker from "react-native-modern-datepicker";

const TaskModal = ({
    modalVisible,
    task,
    setTask,
    handleAddTask,
    handleCancel,
    validationError,
}) => {
    return (
        <Modal
            visible={modalVisible}
            animationType="slide"
            transparent={false}>

            {/* Container for the modal */}
            <View style={styles.modalContainer}>
                <TextInput
                    style={styles.input}
                    placeholder="Title"
                    value={task.title}
                    onChangeText={(text) =>
                        setTask({ ...task, title: text })
                    }
                    // Update the title when text changes
                    />

                <TextInput
                    style={styles.input}
                    placeholder="Description"
                    value={task.description}
                    onChangeText={(text) =>
                        setTask({
                            ...task,
```

```
                            description: text,
                        })
                    }/>

                <Text style={styles.inputLabel}>
                    Deadline:
                </Text>
                <DatePicker
                    style={styles.datePicker}
                    mode="calendar"
                    selected={task.deadline}
                    onDateChange={(date) =>
                        setTask({ ...task, deadline: date })
                    }/>

                {validationError && (
                    <Text style={styles.errorText}>
                        Please fill in all fields correctly.
                    </Text>
                )}
                <Button

                    // Display "Update" when editing an existing
                    // task, "Add" when adding a new task
                    title={task.id ? "Update" : "Add"}
                    // Call the handleAddTask function
                    // when the button is pressed
                    onPress={handleAddTask}
                    // Set the button color
                    color="#007BFF"/>

                <Button
                    title="Cancel"
                    onPress={handleCancel}
                    color="#FF3B30"/>
            </View>
        </Modal>
    );
```

```
};

export default TaskModal;
```

ii. TaskList.js

```
// components/TaskList.js
import React from "react";
import { ScrollView } from "react-native";
import TaskItem from "./TaskItem";
import styles from "../styles";

const TaskList = ({
    tasks,
    handleEditTask,
    handleToggleCompletion,
    handleDeleteTask,
}) => {
    return (
        <ScrollView style={styles.taskList}>

            {/* Scrollable container for the list of tasks */}
            {tasks.map((t) => (
                <TaskItem
                    // Use the task's ID as the key to
                    // uniquely identify each TaskItem
                    key={t.id}

                    // Pass the task object as a prop to TaskItem
                    task={t}

                    // Pass functions to handle editing,
                    // toggling completion, and deletion
                    handleEditTask={handleEditTask}
                    handleToggleCompletion={
                        handleToggleCompletion
```

```
                    }
                    handleDeleteTask={handleDeleteTask}
                />
            ))}
        </ScrollView>
    );
};
// Export the TaskList component
export default TaskList;
```

iii. TaskItem.js

```
// components/TaskItem.js
import React from "react";
import { View, Text, TouchableOpacity } from "react-native";
import styles from "../styles";

const TaskItem = ({
    task,
    handleEditTask,
    handleToggleCompletion,
    handleDeleteTask,
}) => {
    return (
        <View style={styles.taskItem}>
            <View style={styles.taskTextContainer}>
                <Text
                    style={[
                        styles.taskText,
                        task.status === "Completed" &&
                            styles.completedTaskText,
                    ]}>
                    {task.title}
                </Text>
                <Text style={styles.taskDescription}>
                    {task.description}
                </Text>
```

```
                <Text style={styles.taskStatus}>
                    Status: {task.status}
                </Text>
                <Text style={styles.taskDeadline}>
                    Deadline: {task.deadline}
                </Text>
                <Text style={styles.taskCreatedAt}>
                    Created: {task.createdAt}
                </Text>
            </View>
            <View style={styles.buttonContainer}>
                <TouchableOpacity
                    onPress={() => handleEditTask(task)}
                    style={[styles.editButton]}>
                    <Text style={styles.buttonText}>
                        Edit
                    </Text>
                </TouchableOpacity>
                <TouchableOpacity
                    onPress={() =>
                        handleToggleCompletion(task.id)
                    }
                    style={[
                        styles.completeButton,
                        task.status === "Completed" &&
                            styles.completedButton,
                    ]}>
                    <Text style={styles.buttonText}>
                        {task.status === "Completed"
                            ? "Pending"
                            : "Completed"}
                    </Text>
                </TouchableOpacity>
                <TouchableOpacity
                    onPress={() =>
                        handleDeleteTask(task.id)
                    }
                    style={[styles.deleteButton]}>
```

```
                    <Text style={styles.buttonText}>
                        Delete
                    </Text>
                </TouchableOpacity>
            </View>
        </View>
    );
};

export default TaskItem;
```

iv. styles.js

```
// styles.js

import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 20,
        backgroundColor: "#f7f7f7",
    },
    title: {
        fontSize: 28,
        fontWeight: "bold",
        marginBottom: 20,
        color: "#333",
        textAlign: "center",
    },
    taskList: {
        flex: 1,
    },
    taskItem: {
        flexDirection: "row",
        justifyContent: "space-between",
        alignItems: "center",
        backgroundColor: "#fff",
```

```
        marginBottom: 10,
        padding: 15,
        borderRadius: 10,
    },
    taskTextContainer: {
        flex: 1,
    },
    taskText: {
        fontSize: 18,
        fontWeight: "bold",
        color: "#333",
    },
    completedTaskText: {
        textDecorationLine: "line-through",
        color: "gray",
    },
    taskDescription: {
        fontSize: 16,
        color: "#666",
    },
    taskTime: {
        fontSize: 14,
        color: "#666",
    },
    taskStatus: {
        fontSize: 16,
        color: "#666",
    },
    buttonContainer: {
        // Or 'row' depending on your layout
        flexDirection: "column",
        // Adjust the value as needed for the
        // desired spacing
        marginVertical: 2,
    },

    editButton: {
        backgroundColor: "#007BFF",
```

```
        borderRadius: 5,
        padding: 10,
        marginRight: 10,
        width: 110,
    },
    button: {
        marginBottom: 10,
    },
    completeButton: {
        backgroundColor: "#4CAF50",
        borderRadius: 5,
        padding: 10,
        marginRight: 10,
        width: 110,
    },
    completedButton: {
        backgroundColor: "#808080",
    },
    buttonText: {
        color: "#fff",
        fontSize: 15,
    },
    deleteButton: {
        backgroundColor: "#FF9500",
        borderRadius: 5,
        padding: 10,
        width: 110,
    },
    addButton: {
        alignItems: "center",
        justifyContent: "center",
        backgroundColor: "#007BFF",
        paddingVertical: 15,
        borderRadius: 10,
        marginTop: 20,
    },
    addButtonText: {
        color: "#fff",
```

```
        fontSize: 18,
        fontWeight: "bold",
    },
    modalContainer: {
        flex: 1,
        padding: 20,
        backgroundColor: "#fff",
    },
    input: {
        borderWidth: 1,
        borderColor: "#ccc",
        padding: 10,
        marginBottom: 20,
        borderRadius: 5,
        fontSize: 16,
    },
    inputLabel: {
        fontSize: 16,
        fontWeight: "bold",
    },
    errorText: {
        color: "#FF3B30",
        fontSize: 16,
        marginBottom: 10,
    },
    taskDeadline: {
        color: "#FF3B12",
    },
    taskCreatedAt: {
        color: "#5497FF",
    },
});

export default styles;
```

v. App.js

```
// App.js

// Import necessary modules and components
import React, { useState } from "react";
import {
    View,
    Text,
    TouchableOpacity,
    Modal,
    Button,
} from "react-native";

// Import TaskList component
import TaskList from "./components/TaskList";

// Import TaskModal component
import TaskModal from "./components/TaskModel";
import styles from "./styles"; // Import styles

// Define the main App component
const App = () => {
    // Define state variables

    // Array to store tasks
    const [tasks, setTasks] = useState([]);
    const [task, setTask] = useState({
        title: "",
        description: "",
        status: "Pending",
        deadline: "",
        createdAt: "",
    });

    // Task object for creating/editing tasks

    // Modal visibility
    const [modalVisible, setModalVisible] = useState(false);
```

```
// Task being edited
const [editingTask, setEditingTask] = useState(null);
const [validationError, setValidationError] =
    useState(false); // Validation flag

// Function to add a new task or update an existing task
const handleAddTask = () => {
    if (
        task.title.trim() !== "" &&
        task.deadline !== ""
    ) {
        const currentDate = new Date();
        const formattedDate =
            currentDate.toLocaleString();

        if (editingTask) {

            // If editing an existing task, update it
            const updatedTasks = tasks.map((t) =>
                t.id === editingTask.id
                    ? { ...t, ...task }
                    : t
            );
            setTasks(updatedTasks);
            setEditingTask(null);
        } else {

            // If adding a new task, create it
            const newTask = {
                id: Date.now(),
                ...task,

                // Set the creation date and time as a string
                createdAt: formattedDate,
            };
            setTasks([...tasks, newTask]);
        }
```

```
            // Clear the task input fields and reset state
            setTask({
                title: "",
                description: "",
                status: "Pending",
                deadline: "",
                createdAt: "",
            });

            // Close the modal
            setModalVisible(false);

            // Reset validation error
            setValidationError(false);
        } else {

            // Show validation error if fields are not filled
            setValidationError(true);
        }
    };

    // Function to handle task editing
    const handleEditTask = (task) => {

        // Set the task being edited
        setEditingTask(task);

        // Pre-fill the input with task data
        setTask(task);

        // Open the modal for editing
        setModalVisible(true);
    };

    // Function to delete a task
    const handleDeleteTask = (taskId) => {
        const updatedTasks = tasks.filter(
```

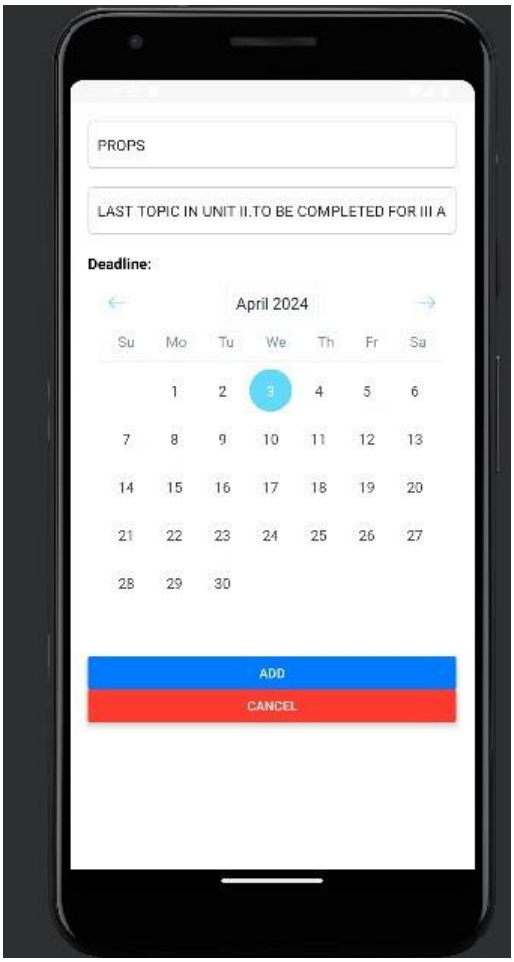
```
            (t) => t.id !== taskId
        );
        setTasks(updatedTasks);
    };

    // Function to toggle task completion status
    const handleToggleCompletion = (taskId) => {
        const updatedTasks = tasks.map((t) =>
            t.id === taskId
                ? {
                    ...t,
                    status:
                        t.status === "Pending"
                            ? "Completed"
                            : "Pending",
                }
                : t
        );
        setTasks(updatedTasks);
    };

    // Return the JSX for rendering the component
    return (
        <View style={styles.container}>
            <Text style={styles.title}>Task Manager</Text>
            {/* Render the TaskList component */}
            <TaskList
                tasks={tasks}
                handleEditTask={handleEditTask}
                handleToggleCompletion={
                    handleToggleCompletion
                }
                handleDeleteTask={handleDeleteTask}
            />
            {/* Button to add or edit tasks */}
            <TouchableOpacity
                style={styles.addButton}
                onPress={() => {
```

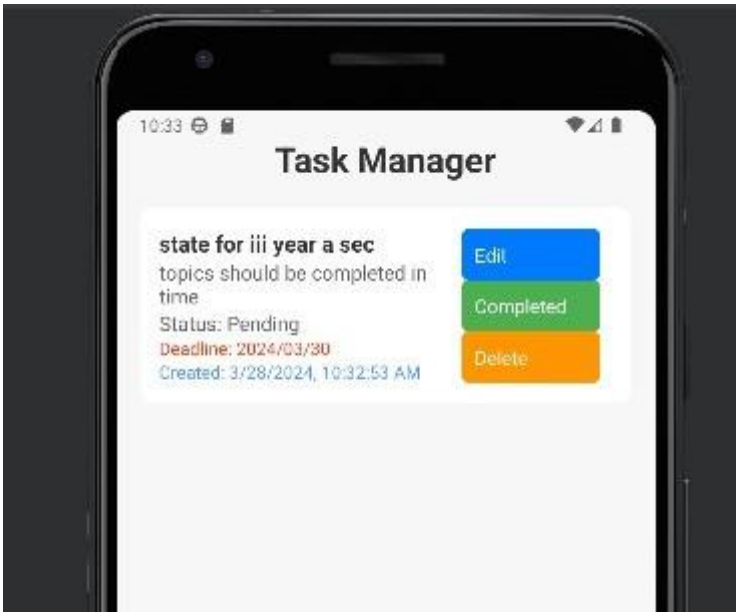
```
                    setEditingTask(null);
                    setTask({
                        title: "",
                        description: "",
                        status: "Pending",
                        deadline: "",
                        createdAt: "",
                    });
                    setModalVisible(true);
                    setValidationError(false);
                }}>
                <Text style={styles.addButtonText}>
                    {editingTask ? "Edit Task" : "Add Task"}
                </Text>
            </TouchableOpacity>
            {/* Render the TaskModal component */}
            <TaskModal
                modalVisible={modalVisible}
                task={task}
                setTask={setTask}
                handleAddTask={handleAddTask}
                handleCancel={() => {
                    setEditingTask(null);
                    setTask({
                        title: "",
                        description: "",
                        status: "Pending",
                        deadline: "",
                        createdAt: "",
                    });
                    setModalVisible(false);
                    setValidationError(false);
                }}
                validationError={validationError}/>
        </View>
    );
};
```

```
// Export the App component as the default export
export default App;
```

Output:

i) Add Task





Result:

Thus a cross platform application has been successfully developed and the outputs are verified.

EX.NO.8 LOCATION TRACKER USING CORDOVA

Date:

Aim:

To design and develop an android application using Apache Cordova to find and display the current location of the user.

Steps:

1. Create a new cordova project
   cordova create loc_geo com.example.location locationgeo
2. Navigate to the Project Directory
   cd loc_geo
3. Add Platforms
   cordova platform add android
4. Add geolocation plugin using
   cordova plugin add cordova-plugin-geolocation
5. Open visual studio code select the folder named loc_geo and open it.
6. Add the code in index.html and index.js.
7. Execute the app
   Run->Run without Debugging->Chrome

Code

i) index.html

```
<html>

     <body>

         <div class="app">

            <h1>GEO LOCATION OF USER</h1>

            <div id="deviceready" class="blink">

               <button id = "getPosition">CURRENT POSITION</button>

<button id = "watchPosition">WATCH POSITION</button>
```

```
            </div>
        </div>
        <script src="cordova.js"></script>
        <script src="js/index.js"></script>
    </body>
</html>
```

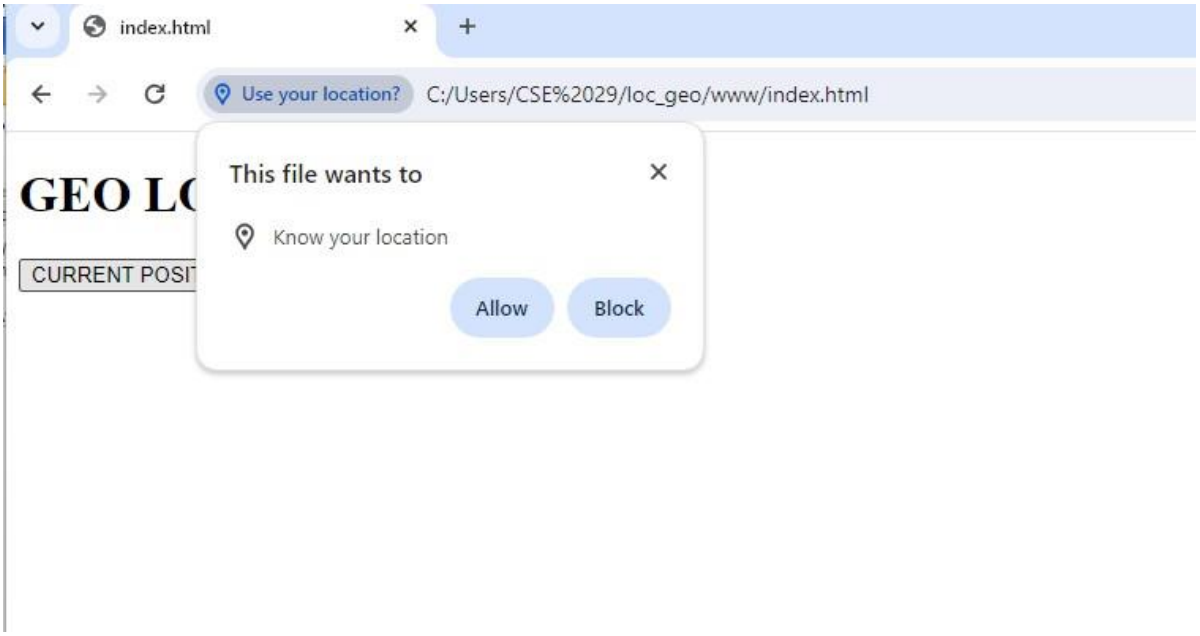
ii) index.js

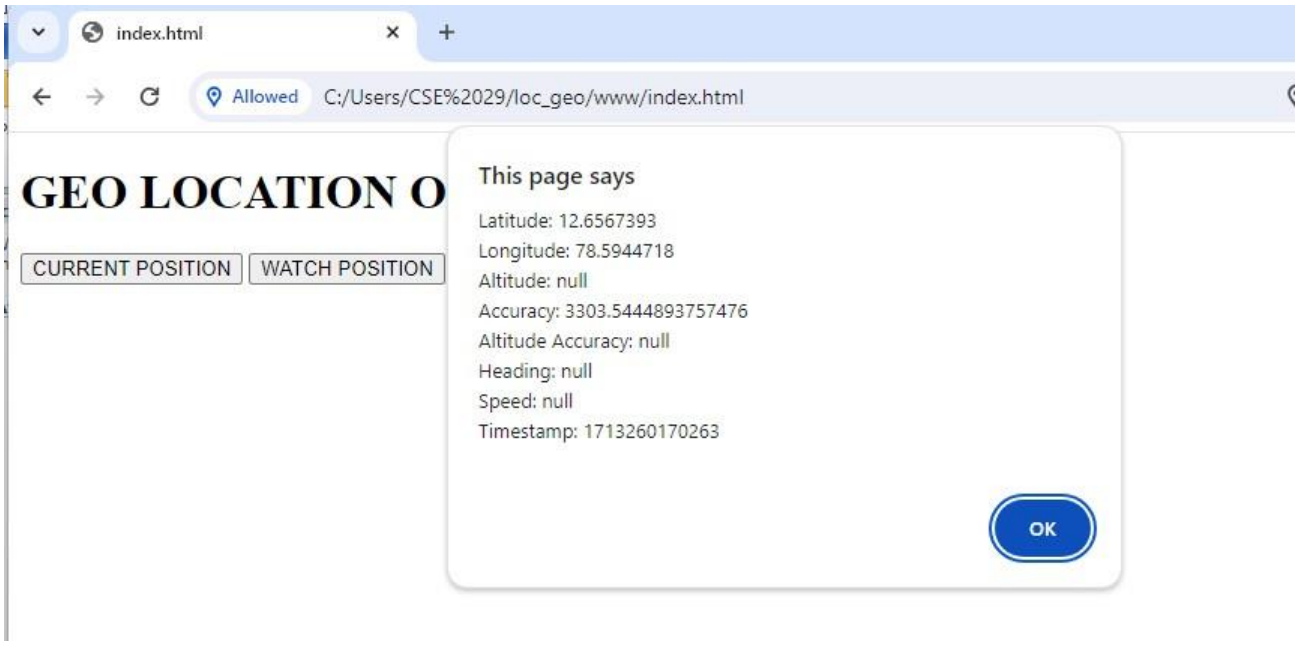
```
document.getElementById("getPosition").addEventListener("click",  getPosition);
document.getElementById("watchPosition").addEventListener("click",
watchPosition);
function getPosition() {
    var options = {
       enableHighAccuracy: true,
       maximumAge: 3600000
    }
    var watchID = navigator.geolocation.getCurrentPosition(onSuccess, onError,
options);

    function onSuccess(position) {
       alert('Latitude: '          + position.coords.latitude          + '\n'
+
          'Longitude:  '       +  position.coords.longitude       +  '\n'  +
          'Altitude:  '        +  position.coords.altitude        +  '\n'  +
          'Accuracy:  '        +  position.coords.accuracy        +  '\n'  +
          'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
```

```
          'Heading: '           + position.coords.heading           + '\n' +
          'Speed: '             + position.coords.speed             + '\n' +
          'Timestamp: '         + position.timestamp                + '\n');
    };

    function onError(error) {
       alert('code: '    + error.code    + '\n' + 'message: ' + error.message
+ '\n');
    }
 }

 function watchPosition() {
    var options = {
       maximumAge: 3600000,
       timeout: 3000,
       enableHighAccuracy: true,
    }
    var watchID = navigator.geolocation.watchPosition(onSuccess, onError,
options);

    function onSuccess(position) {
       alert('Latitude: '          + position.coords.latitude          + '\n'
+
          'Longitude: '         + position.coords.longitude         + '\n' +
          'Altitude: '          + position.coords.altitude          + '\n' +
          'Accuracy: '          + position.coords.accuracy          + '\n' +
```

```
          'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
          'Heading: '           + position.coords.heading           + '\n' +
          'Speed: '             + position.coords.speed              + '\n' +
          'Timestamp: '         + position.timestamp                 + '\n');
    };


    function onError(error) {
       alert('code: '    + error.code    + '\n' +'message: ' + error.message +
'\n');
    }
 }
```

Output:





Result:

Thus an application for finding the location of the user has been successfully developed and the outputs are verified.

EX.NO.9 # LOGIN FORM USING CORDOVA

Date:

Aim:

To design an android application using Cordova for a user login screen with username,password and a submit button.

Steps:

1. Install cordova

   npm install –g cordova

2. **Add the following,if system is restricted to policy,**

   Set-ExecutionPolicy –Executionpolicy Unrestricted –Scope CurrentUser

3. Create a new cordova project

   cordova create LoginAppli com.example.loginappli LoginAppli

4. Navigate to the Project Directory

   cd LoginAppli

5. Add Platforms

   cordova platform add android

   Open visual studio code select the folder named **LoginAppli** and open it.

6. Add the code in index.html and index.js.
7. Execute the app

   Run->Run without Debugging->Chrome

Code

i) index.html

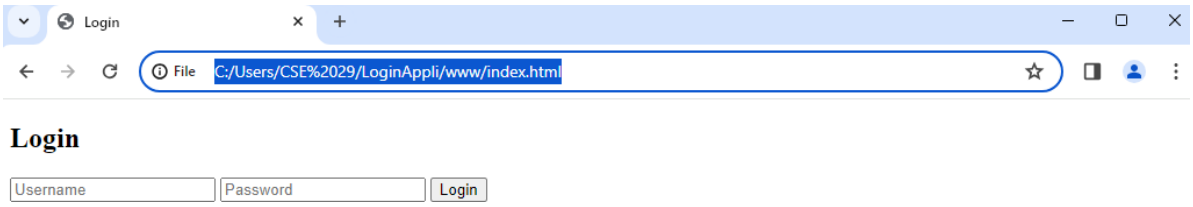
```
<html>

<head>

    <title>Login</title>

    <link rel="stylesheet" type="text/css" href="css/style.css">

</head>

<body>

    <div class="login-container">

        <h2>Login</h2>

        <input type="text" id="username" placeholder="Username">

        <input type="password" id="password" placeholder="Password">

        <button onclick="login()">Login</button>

        <p id="error-message"></p>

    </div>

    <script src="js/index.js"></script>

</body>

</html>
```

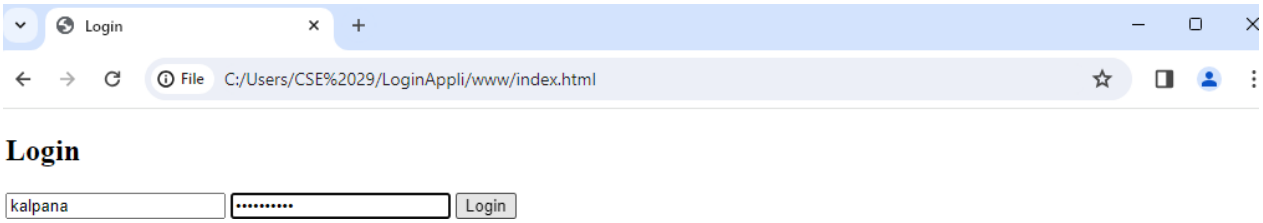
ii) index.js

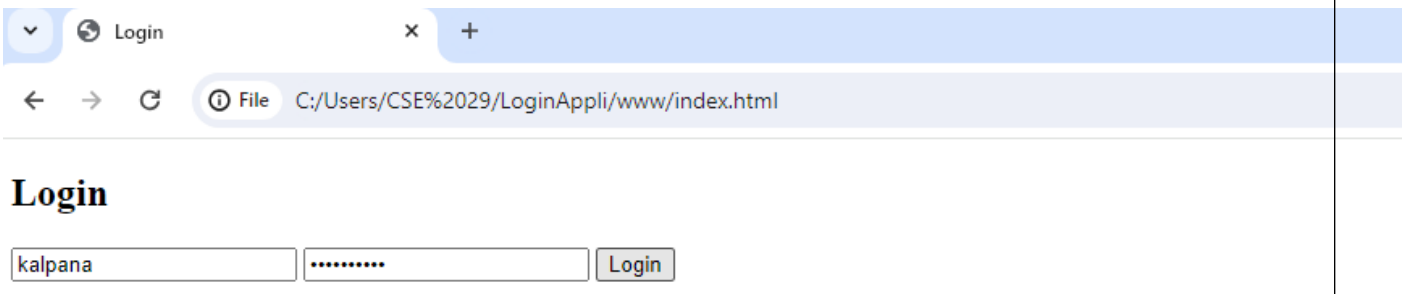
```
function login() {

    var username = document.getElementById('username').value;

    var password = document.getElementById('password').value;
```

```
    // Perform your login authentication here, for example:
    if (username === 'admin' && password === 'password') {
        // Redirect to another page upon successful login
        window.location.href = 'dashboard.html';
    } else {
        // Display error message
        document.getElementById('error-message').innerHTML = 'Invalid username or password';
    }
}
```

Output:


Login
File
C:/Users/CSE%2029/LoginAppli/www/index.html
Login
Username
Password
Login


Login
File
C:/Users/CSE%2029/LoginAppli/www/index.html
Login
kalpana
Login


Login
File
C:/Users/CSE%2029/LoginAppli/www/index.html
Login
kalpana
Login
Invalid username or password

Result:

Thus an application for login screen using cordova has been successfully developed and the outputs are verified.